3. Использование сигналов

В этом уроке рассмотрим сигналы. Это сообщения, которые выдает узел при определенных событиях, например, при нажатии кнопки. Другие узлы могут получать этот сигнал и вызывать функции, соответствующие событию.

Сигналы это механизм делегирования, встроенный в Godot, который позволяет одному игровому объекту реагировать на изменение в другом без их привязки друг к другу. Использование сигналов снижает связывание и делает код более гибким.

Например, на экране может быть полоса жизни, которая отображает здоровье игрока. Когда игрок получает урон или использует лечебное зелье, вы хотите, чтобы полоса отражала изменение. Для этого в Godot вы использовали бы сигналы.

Как и методы, сигналы являются типом первого класса с Godot 4.0. Это означает, что вы можете передавать их как аргументы метода напрямую, без необходимости передавать их как строки, что обеспечивает лучшее автодополнение и менее подвержено ошибкам.

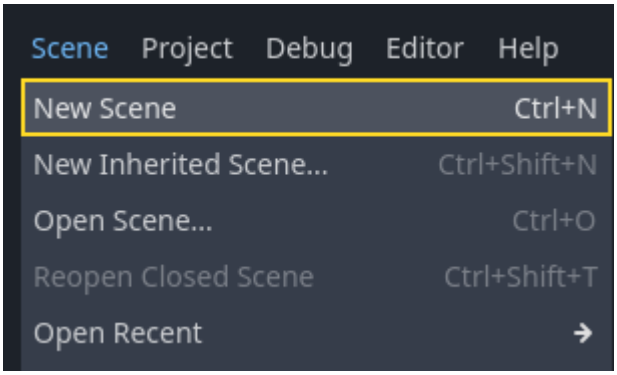
Теперь мы будем использовать сигнал, чтобы заставить нашу иконку Godot из предыдущего урока двигаться и останавливаться при нажатии кнопки.

> Для этого проекта мы будем следовать правилам именования Godot.
>
> - **GDScript**: Классы (узлы) используют PascalCase, переменные и функции - snake_case, константы - ALL_CAPS.

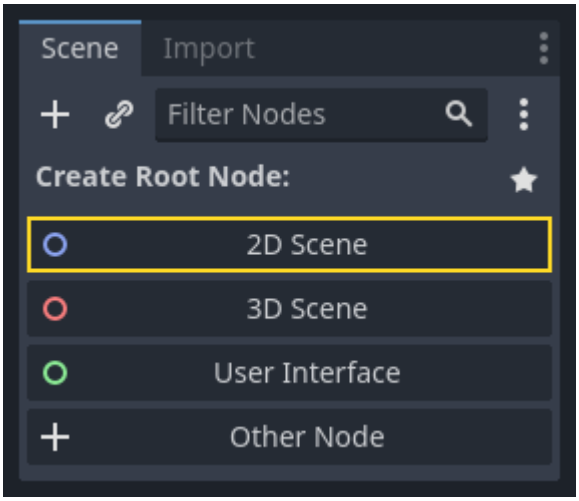
Настройка сцены

Чтобы добавить кнопку в нашу игру, мы создадим новую сцену, которая будет включать как кнопку, так и сцену `sprite_2d.tscn`, которую мы создали в уроке Создание вашего первого скрипта.

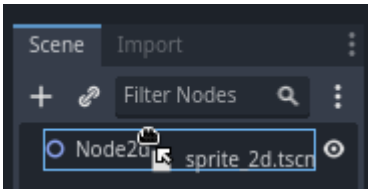
Создайте новую сцену через меню Сцена -> Новая сцена.



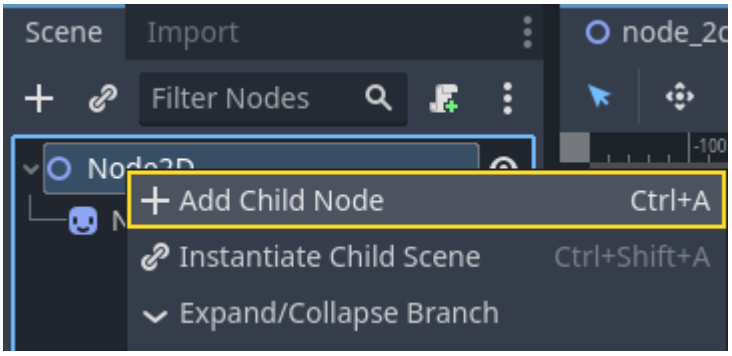
На панели Сцена (Scene) нажмите кнопку "2D сцена" (2D Scene). Это добавит Node2D в роли корневой сцены.



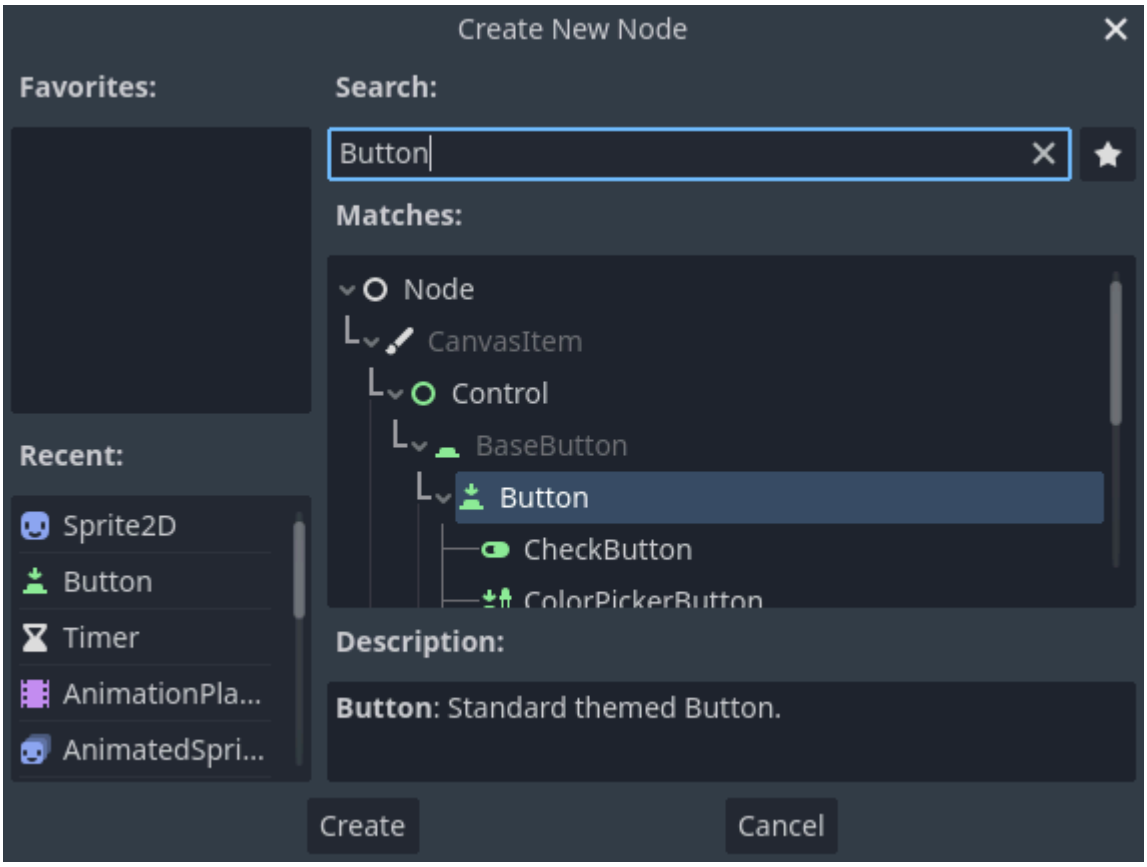
На панели "Файловая система" (File System) зажмите и перетащите файл `sprite_2d.tscn`, который вы сохранили ранее, на узел Node2D, чтобы создать экземпляр его класса.



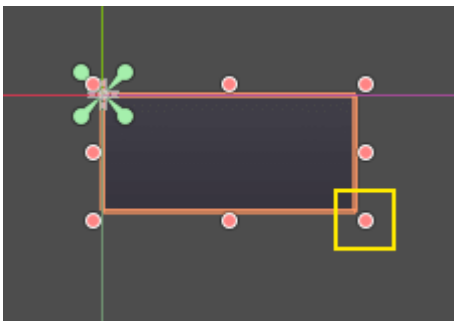
Мы хотим добавить другой узел рядом с Sprite2D. Для этого нажмите ПКМ на Node2D и выберите Добавить дочерний узел.



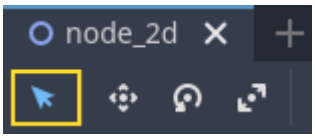
Найдите узел Button и добавьте его.



По умолчанию этот узел небольшой. Зажмите и потяните правый нижний маркер кнопки в окне просмотра, чтобы изменить его размер.

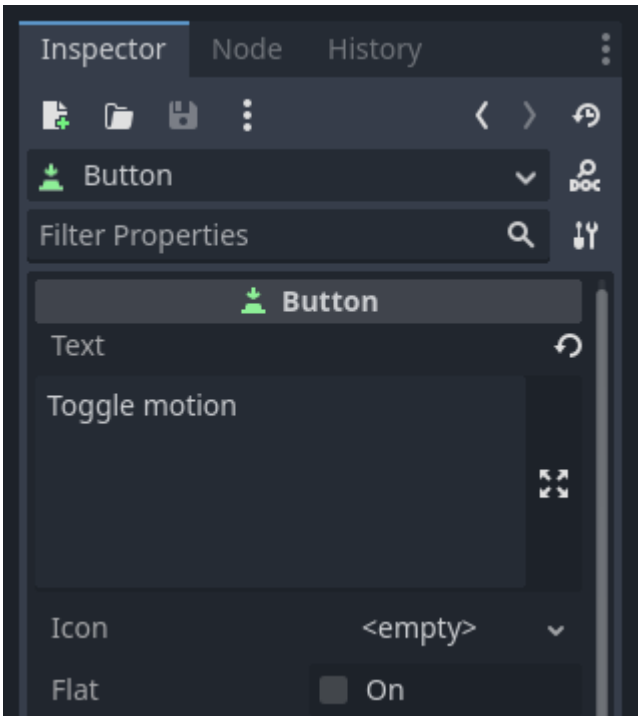


Если вы не видите маркеров, убедитесь, что на панели инструментов активен инструмент выделения.

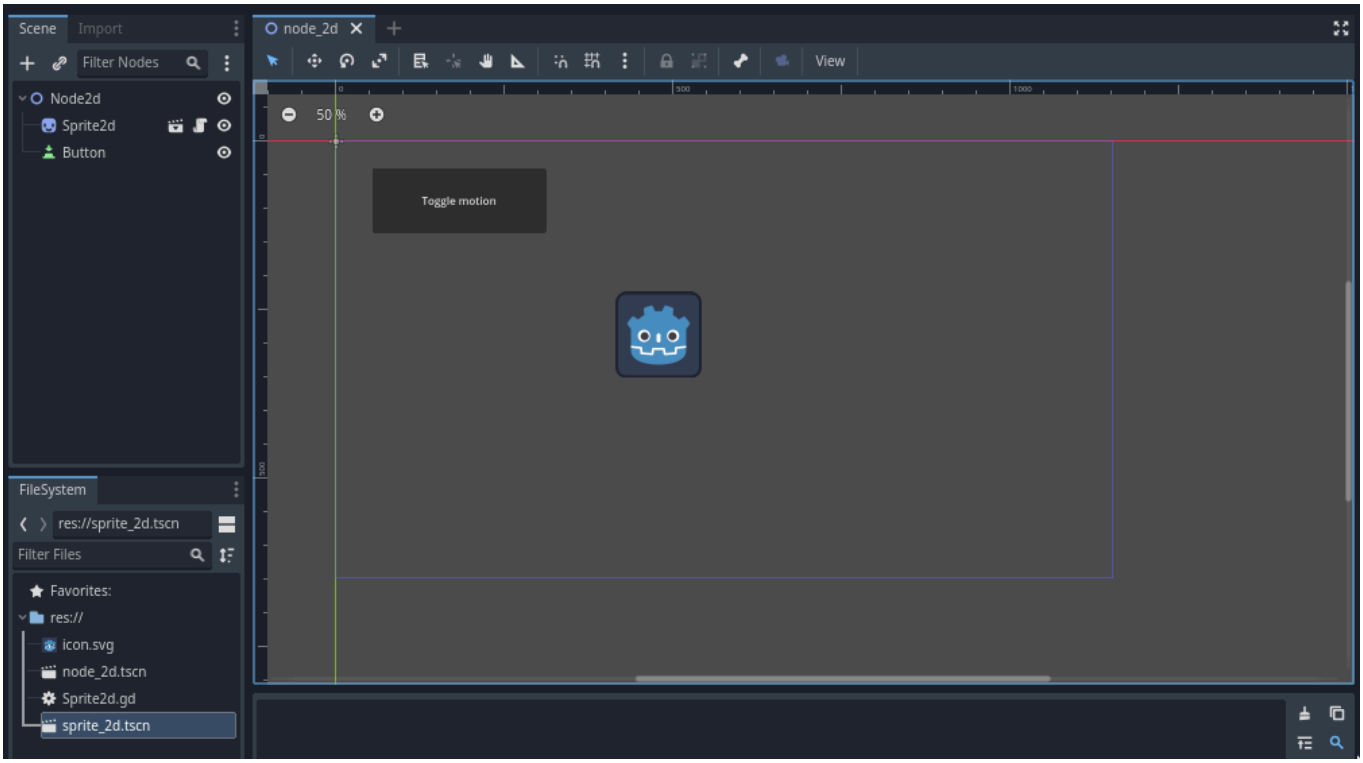


Зажмите и перетащите саму кнопку, чтобы подвинуть её ближе к спрайту.

Вы также можете сделать надпись на кнопке, отредактировав свойство Text в Инспекторе. Введите `Toggle motion`.



Дерево сцены и область просмотра должны выглядеть следующим образом.



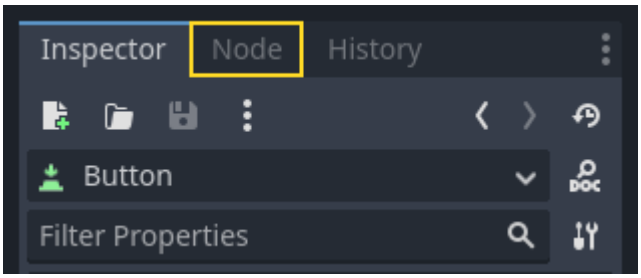
Сохраните вашу только что созданную сцену как `node_2d.tscn`, если этого ещё не сделали. Вы можете запустить её с помощью F6. Теперь будет видна кнопка, но если вы её нажмёте, ничего не произойдет.

Подключение сигнала в редакторе

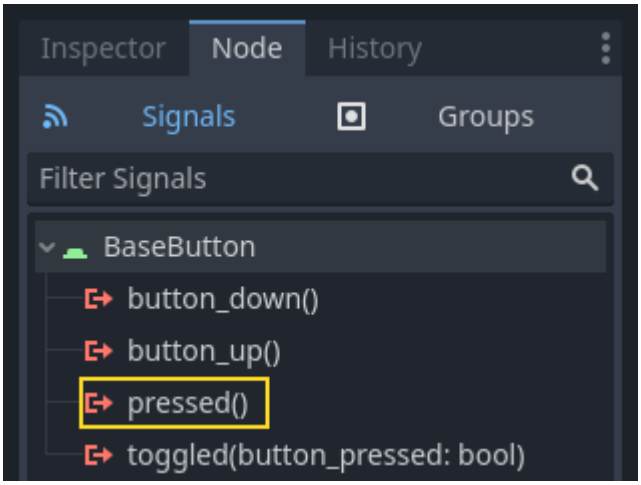
Теперь мы хотим присоединить сигнал "нажата" (pressed) кнопки (Button) к нашему спрайту (Sprite2D), и мы хотим вызвать новую функцию, которая будет включать и

отключать его передвижение. Нам нужно, чтобы скрипт, который мы писали на предыдущем уроке, был прикреплён к спрайту.

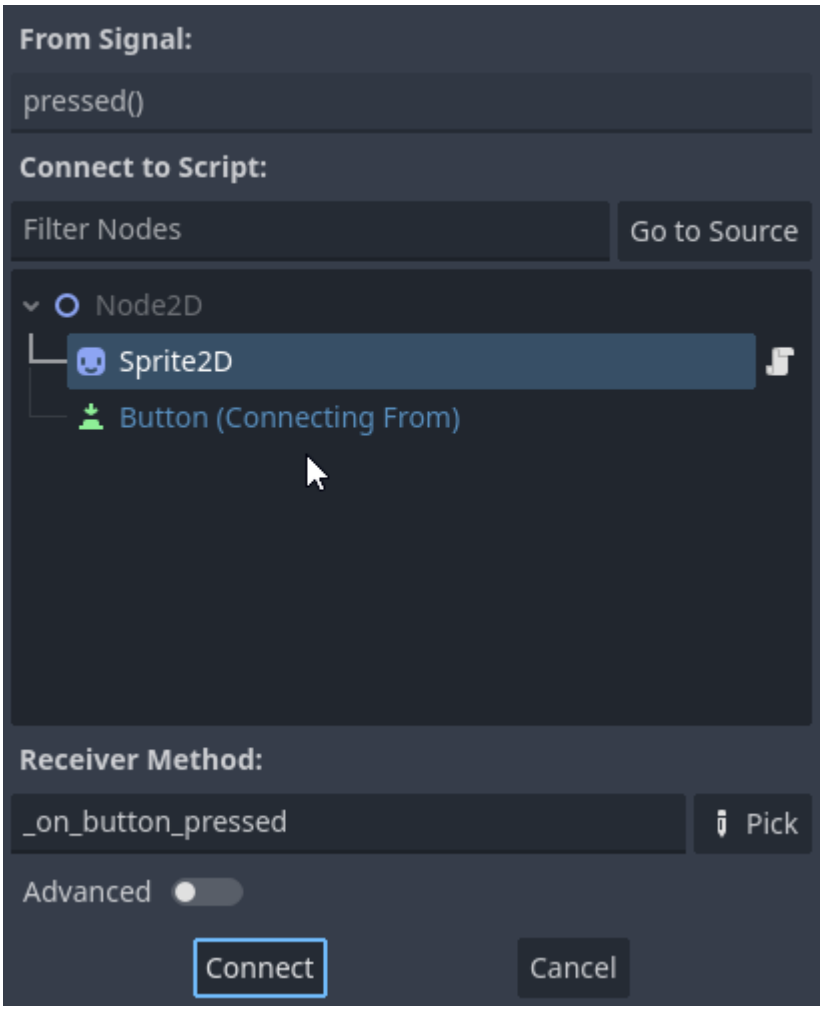
Вы можете подключить сигналы в панели Узел. Выберите узел Button и на правой стороне экрана нажмите вкладку "Узел" рядом с Инспектором.



На панели отображаются сигналы, доступные выбранному узлу.



Дважды кликните сигнал "pressed", откроется окно присоединения узлов.

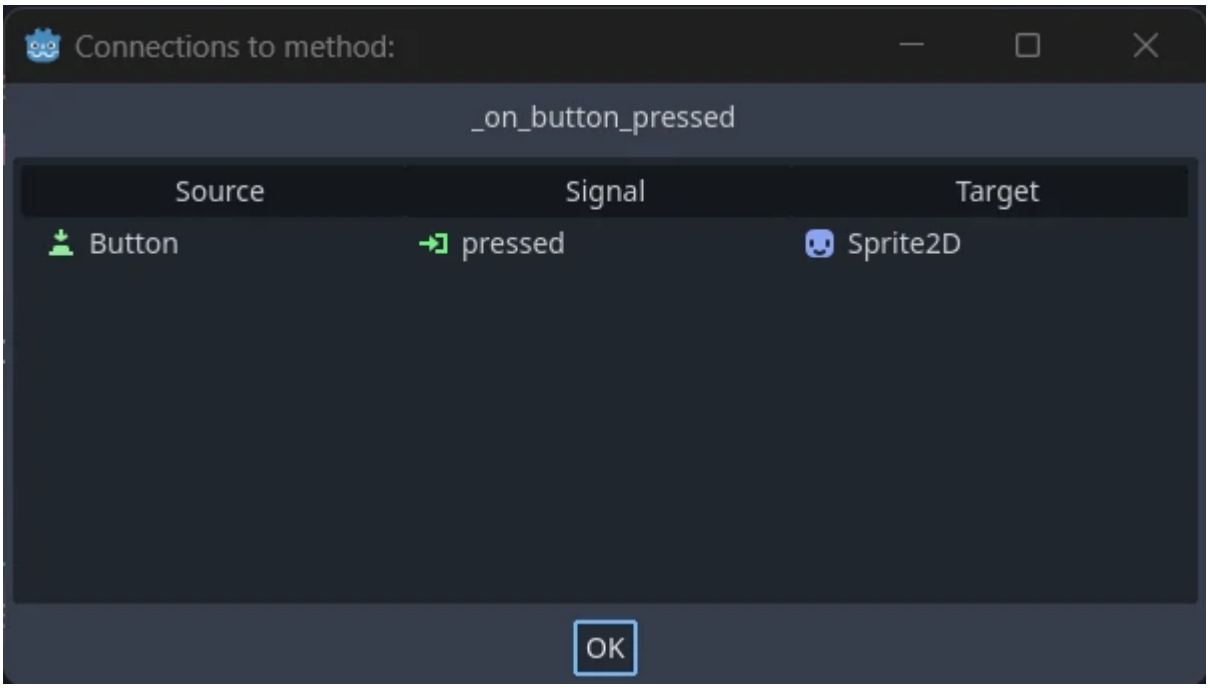


Теперь вы можете подключить сигнал к узлу Sprite2D . Узлу нужна функция, которую вызовет Godot, когда Button выдаст сигнал. Редактор самостоятельно создаст её. По соглашению, мы называем эти обратные вызовы "_on_NodeName_signal_name". У нас это будет "_on_Button_pressed".

Нажмите кнопку "Connect" ("Присоединить") для завершения соединения и перейдите к редактированию скрипта. Вы увидите новый метод (функцию) с иконкой соединения на левой границе.

```
23 func _on_button_pressed():
24     pass # Replace with function body.
25
```

Если нажать иконку, всплывёт окно с информацией о соединении. Это доступно только при присоединении узлов в редакторе.



Давайте заменим строку со словом `pass` кодом, который изменяет движение узла. Наш Sprite2D движется благодаря коду функции `_process()`. Godot предоставляет метод для включения и выключения обработки: Node.set_process(). Другой метод класса Node, `is_processing()`, возвращает `true`, если обработка активна. Мы можем использовать ключевое слово `not` для инвертирования значения.

```
func _on_button_pressed():
        set_process(not is_processing())
```

Эта функция будет переключать обработку и, в том числе, движение значка по нажатию кнопки.

Перед тем, как попробовать поиграть, нам необходимо упростить нашу функцию `_process()`, чтобы движение узла было автоматическим и не ожидало команд пользователя. Замените текущий код функции на тот, который мы видели два урока назад:

```
func _process(delta):
        rotation += angular_speed * delta
        var velocity = Vector2.UP.rotated(rotation) * speed
        position += velocity * delta
```

Ваш полный код `sprite_2d.gd` должен выглядеть следующим образом.

```
extends Sprite2D

var speed = 400
```

```
var angular_speed = PI


func _process(delta):
        rotation += angular_speed * delta
        var velocity = Vector2.UP.rotated(rotation) * speed
        position += velocity * delta


func _on_button_pressed():
        set_process(not is_processing())
```

Запустите сцену и нажмите кнопку, чтобы увидеть запуск и остановку спрайта.

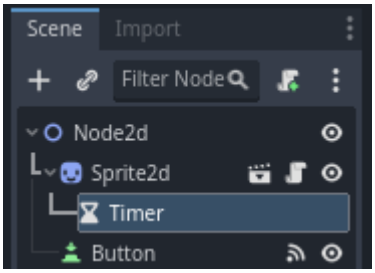
Подключение сигналов в коде

Вы можете присоединять сигналы в коде вместо использования редактора. Это нужно, когда узлы или элементы сцены создаются в скрипте.

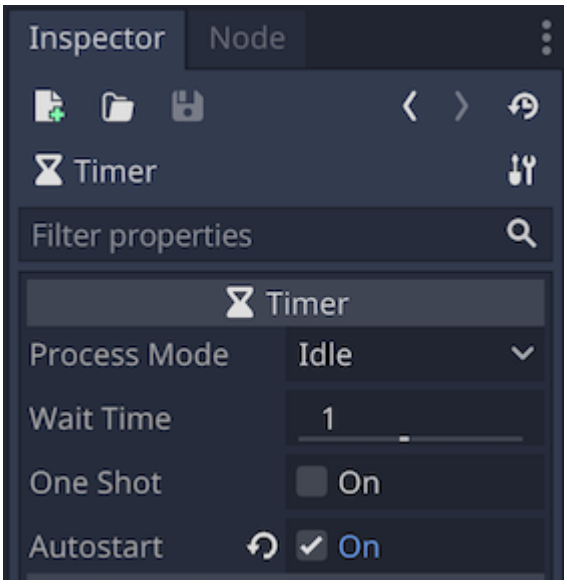
Давайте теперь используем различные узлы. У Godot есть узел Timer, который полезен для реализации задержки перезарядки способностей, перезарядки оружия и другого.

Вернитесь в рабочее пространство 2D. Вы можете нажать на текст «2D» в верхней части окна или нажать Ctrl + F1.

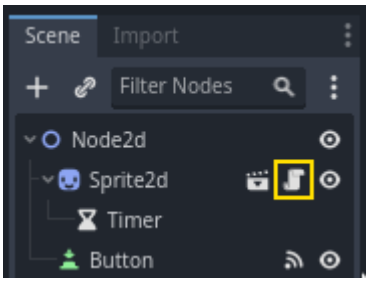
Нажмите правой кнопкой мыши на панели Сцена (Scene) на узел Спрайт (Sprite2D) и добавьте дочерний узел. Найдите Таймер (Timer) и добавьте соответствующий узел. Теперь ваша сцена должна выглядеть так.



Выбрав узел Таймера, откройте Инспектор и активируйте свойство **Autostart**.



Нажмите на иконку скрипта рядом со спрайтом (Sprite2D), чтобы вернуться к рабочей области скрипта.



Нам необходимо выполнить две операции для подключения узлов через код:

1. Получить ссылку на таймер (Timer) из спрайта (Sprite2D).
2. Вызвать метод `connect()` для сигнала "timeout" таймера (Timer).

> Нам необходимо выполнить две операции для подключения узлов через код:
>
> 1. Получить ссылку на таймер (Timer) из спрайта (Sprite2D).
> 2. Вызвать метод `connect()` для сигнала "timeout" таймера (Timer).

Мы хотим подключить сигнал, когда сцена инстансируется, и мы можем это сделать, используя встроенную функцию `Node.ready()`, которая автоматически вызывается движком, когда узел полностью инстансирован.

Чтобы получить ссылку на узел, относящийся к действующему узлу, мы используем метод `Node.get_node()`. Мы можем сохранить ссылку в переменной.

```
func _ready():
        var timer = get_node("Timer")
```

Функция `get_node()` смотрит на дочерние узлы спрайта (Sprite2D) и получает узлы по их имени. К примеру, если Вы переименуете в редакторе узел Таймера в "BlinkingTimer", то Вам нужно будет поменять вызов на `get_node("BlinkingTimer")`.

Сейчас мы можем подключить таймер (Timer) к спрайту (Sprite2D) в функции `_ready()`.

```
func _ready():
        var timer = get_node("Timer")
        timer.timeout.connect(_on_timer_timeout)
```

Строка читается так: мы подключаем сигнал Таймера "timeout" к узлу, к которому прикреплён текущий скрипт. Когда Таймер подаёт сигнал `timeout`, мы хотим вызвать функцию `_on_timer_timeout()`, которую нам необходимо определить. Давайте добавим её в нижней части нашего скрипта и используем её для переключения видимости нашего спрайта.

```
func _on_timer_timeout():
        visible = not visible
```

Свойство `visible` имеет логический тип (boolean), оно контролирует видимость нашего узла. Строка `visible = not visible` переключает значение. Если `visible` равно `true`, оно станет `false`, и наоборот.

Если вы сейчас запустите сцену Node2D, вы увидите, что спрайт мигает с интервалом в одну секунду.

Готовый скрипт

Вот и всё для нашей маленькой движущейся и мигающей демонстрации с иконкой Godot! Вот полный файл `sprite_2d.gd` для справки.

```
extends Sprite2D

var speed = 400
var angular_speed = PI


func _ready():
        var timer = get_node("Timer")
        timer.timeout.connect(_on_timer_timeout)


func _process(delta):
```

```
        rotation += angular_speed * delta
        var velocity = Vector2.UP.rotated(rotation) * speed
        position += velocity * delta


func _on_button_pressed():
        set_process(not is_processing())


func _on_timer_timeout():
        visible = not visible
```

Пользовательские сигналы

Вы можете определить собственные сигналы в скрипте. Например, вы хотите вывести экран "Конец игры", когда здоровье игрока станет равным нулю. Для этого вы можете определить сигнал "died" или "health_depleted", и выдать его, когда здоровье игрока будет 0.

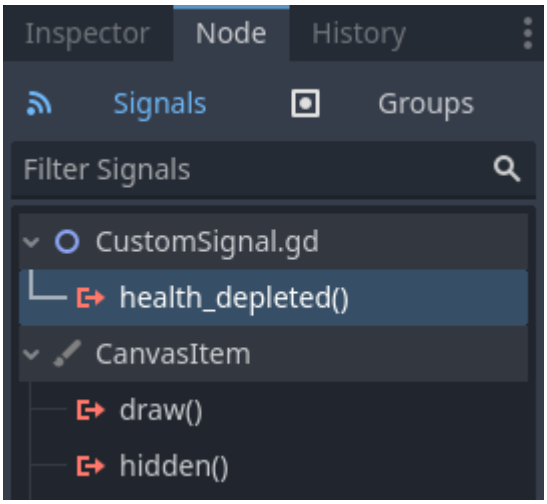
```
extends Node2D

signal health_depleted

var health = 10
```

> Поскольку сигналы представляют собой события, которые только что произошли, мы обычно используем в их названиях глагол действия в прошедшем времени.

Ваши сигналы работают таким же образом, как и встроенные: они появляются во вкладке Узел (Node), и вы можете подключиться к ним как и любым другим.



Чтобы подать сигнал в ваших скриптах, вызовите метод `emit()` в сигнале.

```
func take_damage(amount):
        health -= amount
        if health <= 0:
                health_depleted.emit()
```

Сигнал может дополнительно объявить один или несколько аргументов. Укажите имена аргументов между круглыми скобками:

```
extends Node2D

signal health_changed(old_value, new_value)

var health = 10
```

> Аргументы сигнала показываются в панели Узел редактора, и Godot может использовать их, чтобы производить для вас функции обратного вызова. Однако, вы всё ещё можете отправлять любое число аргументов при отправке сигналов. Это значит, что отправка правильных значений зависит от вас.

Чтобы передать значения вместе с сигналом, добавьте их в качестве дополнительных аргументов к функции `emit()`:

```
func take_damage(amount):
        var old_health = health
        health -= amount
        health_changed.emit(old_health, health)
```

Подведение итогов

Любой узел в Godot излучает сигналы, когда с ним происходит что-то конкретное, как, например, нажатие на кнопку. Другие узлы могут подключиться к отдельным сигналам и среагировать на выбранные события.

Сигналы имеют множество применений. С их помощью Вы можете отреагировать на узел при выходе и входе в игровой мир, на столкновение, на персонажа, входящего или покидающего какую-либо область, на изменение размера элемента интерфейса и на многое другое.

Например, Area2D, представляющий монету, испускает сигнал `body_entered` всякий раз, когда физическое тело игрока входит в его форму столкновения, что позволяет узнать, когда игрок подобрал монету.